UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Creación de una Herramienta de Optimización Bayesiana en Python

Autor: Daniel Fernández Sánchez
Tutor: Daniel Hernández Lobato

junio 2019

Daniel Fernández Sánchez
Creación de una Herramienta de Optimización Bayesiana en Python

Daniel Fernández Sánchez
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Agradecimientos

En primer lugar, me gustaría dar gracias a todos los profesores que me han ayudado y guiado durante el transcurso de la carrera.

También me gustaría agradecer a mi tutor, Daniel Hernández Lobato, por resolver pacientemente todas las dudas que me iban surgiendo y por intentar darme feedback lo más rápido posible.

Además, quiero agradecer a mi madre, mi padre y mi hermano por prestarme toda la ayuda que iba necesitando, por estar siempre ahí para apoyándome cuando tenía problemas y por nunca dejar de confiar en mí.

Resumen

Los avances producidos en ciencia e ingeniería han permitido la mejora y la elaboración de nuevos productos. Una gran parte de estos han sido obtenidos mediante complejos métodos de diseño, en los que se ha debido tener en cuenta una gran cantidad de factores. Por ejemplo: un grupo de investigadores que estén diseñando un nuevo material para almacenar energía, la síntesis de ese material requeriría hacer lentos y costosos experimentos en el laboratorio, por lo que se quiere minimizar su número. Además, los experimentos podrían contar con ruido, cosa que dificultaría esta tarea. Por lo tanto, en este problema se carece de la expresión analítica de la $función\ objetivo$, es costoso efectuar observaciones y estas pueden estar contaminadas por ruido. Con el propósito de abordar este de tipo de problemas, se podría usar la optimización Bayesiana. Esta técnica de optimización emplea la distribución predictiva de un modelo de la $función\ objetivo$, para calcular la utilidad de realizar una observación. Este cálculo se lleva a cabo mediante la función de adquisición. Al maximizar la función de adquisición se obtiene dónde se espera que sea de mayor utilidad llevar a cabo la siguiente observación, para minimizar el número de pasos a efectuar.

En este trabajo se comienza realizando un estudio de los procesos Gaussianos en regresión, los cuales son el modelo más utilizado para aproximar la $función\ objetivo$. Más adelante se especificará como se mide la utilidad de realizar una observación. Una vez finalizado el marco teórico de la optimización Bayesiana, se hablará de las tecnologías empleadas en su elaboración, y de cómo usar la herramienta implementada. Después, se expondrán los experimentos de optimización realizados. Finalmente se darán las conclusiones obtenidas de este trabajo y posibles vías de trabajo futuro.

Palabras clave

Aprendizaje automático, regresión, optimización Bayesiana, procesos Gaussianos

Abstract

The progress in science and engineering have led to improvements and the development of new products. Most of this have been obtained by complex design methods, in which a large number of factors had to be taken into account. For example: a group of researchers who are designing a new material to store energy, the synthesis of that material would require slow and expensive experiments in the laboratory, thus we want to minimize their number. In addition, the experiments could be noise, which would make this task difficult. Therefore, in this problem there is no analytical expression of the $objective\ function$, it is expensive to make observations and these can be noisy. In order to address such problems, Bayesian optimization could be used. This optimization technique uses the predictive distribution of a model of the $objective\ function$ to calculate the usefulness of making an observation. This calculation is carried out using the acquisition function. The maximization of the acquisition function provides where the next observation is expected to be more useful in order to minimize the number of steps to be taken.

The present study begins with the analysis of the Gaussian processes in regression, which are the model most used to approximate the $objective\ function$. Later on, it will be specified how the usefulness of making an observation is measured. Once the theoretical framework of Bayesian optimization is finished, the technologies used in this elaboration, and how to use the implemented tool will be described. Afterwards, the optimization experiments carried out will be presented. Finally, the conclusions obtained from this work and possible future ways of working will be given.

Keywords

Machine learning, regression, Bayesian optimization, Gaussian processes

Índice

Listas

Lista de algoritmos

Lista de códigos

Lista de ecuaciones

Lista de figuras

1

Introducción

En este capítulo se comenzará especificando cuál es la motivación de este trabajo, después se describirán los objetivos generales y específicos que ha habido en su realización, y finalmente se expondrá la estructura que sigue este documento.

1.1. Motivación

El diseño juega un papel esencial dentro en ciencia e ingeniería, y suele ser un problema difícil de abordar, a causa de la gran cantidad de parámetros que se deben tener en cuenta. Si se quisieran obtener los valores óptimos de dichos parámetros, se podría realizar una búsqueda exhaustiva. Sin embargo, debido a la gran cantidad de posibles configuraciones esto resultaría lento, y no siempre sería posible de llevar a cabo.

En este tipo de problemas no se cuenta con una forma de la $función\ objetivo$, por lo tanto, no es posible prever que parámetros darán mejores resultados. Además, probablemente realizar evaluaciones sea costoso, como puede ser la creación de un nuevo producto, en el cual si se quiere probar se deba llamar a usuarios para que lo testeen. Asimismo, cabe la posibilidad de que las observaciones del problema tengan ruido.

Con la finalidad de resolver este problema se podría utilizar la optimización Bayesiana, que es un método de aprendizaje automático. Este método permite minimizar la cantidad de evaluaciones necesarias a efectuar para encontrar el mínimo (o el máximo) de una función. A fin de lograr este propósito, lleva a cabo dos tareas: la construcción de un modelo que aproxima la realidad en base a unos resultados, y por otro la optimización de la utilidad de realizar nuevas observaciones.

Por lo tanto, la optimización Bayesiana es una estrategia que utiliza el modelo, creado a partir de los datos observados, para realizar la siguiente medición. En consecuencia, la utilización de una herramienta que use esta técnica se presenta como una opción superior a las que realizan búsqueda uniforme.

1.2. Objetivos

El objetivo general de este trabajo, es el desarrollo de una herramienta de optimización Bayesiana, que permitiese encontrar unos parámetros superiores, y en menos evaluaciones que los que se encontraría realizando una búsqueda aleatoria o en rejilla, para una $función\ objetivo$. A fin de lograr este propósito, fue necesario cumplir con los siguientes objetivos específicos:

* Estudiar los procesos Gaussianos en regresión, puesto que serían utilizados para realizar predicciones, en base a unas evidencias y una creencia a priori del funcionamiento de los datos.
* Estudiar como la optimización Bayesiana es capaz de minimizar la cantidad de observaciones a realizar, mediante el uso de un modelo que aproxime la $función\ objetivo$ y la cuantificación de la utilidad de realizar una observación.
* Implementar una herramienta con el algoritmo de optimización Bayesiana, usando el framework GPy para la manipulación de los procesos Gaussianos.
* Llevar a cabo experimentos con la herramienta, para probar su funcionamiento en diferentes problemas de optimización.

1.3. Organización de la memoria

Además de la introducción, esta memoria cuenta con la siguiente división por capítulos:

Procesos Gaussianos En este capítulo se comenzará con una introducción a los procesos Gaussianos y se especificará como con ellos es posible realizar predicciones.

Optimización Bayesiana En este capítulo se podrá ver cómo es posible minimizar el número de elecciones a realizar en una $función\ objetivo$, mediante el uso de la optimización Bayesiana.

Herramienta de optimización Bayesiana En este capítulo se describirán las tecnologías utilizadas en el desarrollo de una herramienta de optimización Bayesiana y como utilizar dicha herramienta.

Experimentos realizados En este capítulo se expondrán los resultados obtenidos por la herramienta en dos experimentos con datos sintéticos y dos experimentos con datos reales.

Conclusiones y trabajo futuro En este capítulo se recogerán las principales conclusiones extraídas del uso de la herramienta y se especificarán posibles vias de trabajo futuro.

2

Procesos Gaussianos

Los problemas que conciernen al área del aprendizaje automático comúnmente son divididos en dos tipos, los de regresión y los de clasificación. En los del primer tipo se obtiene una función de regresión, que puede tomar valores en todo $\mathbb{R}$, mientras que en clasificación, los valores están determinados por un conjunto finito de etiquetas. En este TFG se trabajará en el ámbito de los problemas de regresión. Este tipo de problemas se pueden definir como $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1, ..., n\}$, donde $\mathcal{D}$ es un conjunto de datos, que está compuesto por un conjunto de n pares $\mathbf{x}_i$ e y_i, que representan las entradas y los resultados, respectivamente.

Se podría intentar abordar este problema de predicción utilizando solo funciones lineales. Sin embargo, este tipo funciones generalmente carecen de la suficiente complejidad, o flexibilidad, para describir conjuntos de datos complejos. Por otro lado, al utilizar funciones con más grados de libertad, por ejemplo, polinomiales, a veces resultan demasiado complejas, y como resultado aprenden características específicas del conjunto de datos de entrenamiento no generalizables al conjunto de test. Este suceso es conocido como sobreaprendizaje, sobreajuste u $overfitting$. Aunque es posible reducir el sobreaprendizaje aumentando la cantidad de datos, no siempre es posible conseguir la cantidad suficiente, por lo que este enfoque no suele ser una buena opción [1, Capítulo 1]. Por otra parte, si se realizase una aproximación donde el modelo a utilizar no tuviera en cuenta dicha complejidad para representarlos, sino que pudiese adaptarse a medida que aumenta el número de ellos y además fuese robusto al posible ruido que pudiesen tener, entonces, se estaría pensando en una solución con las características de los procesos Gaussianos.

2.1. Introducción a los procesos gaussianos

En los problemas de regresión abordados en este trabajo se han usado los procesos Gaussianos, aunque también se pueden utilizar en problemas de clasificación. La explicación de esta introducción se hará siguiendo [2, Capítulo 1]. Dichos procesos Gaussianos se obtienen de la generalización de la distribución de probabilidad Gaussiana. La ecuación que define esta distribución para una dimensión viene dada por la siguiente expresión

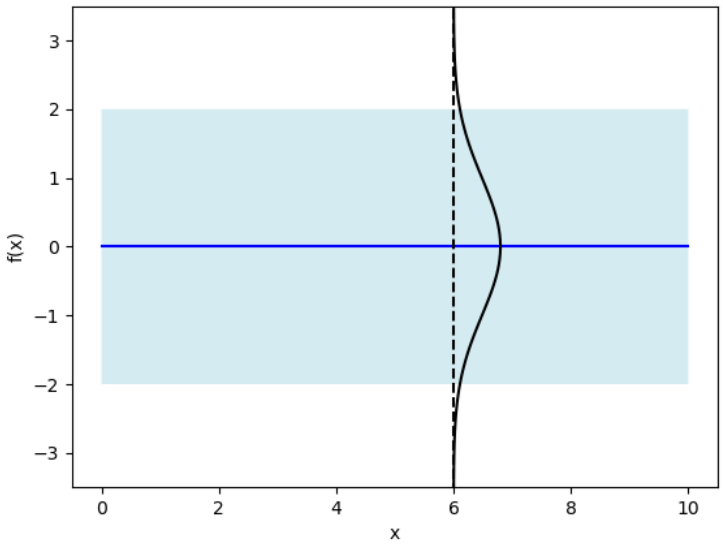


Figura 2.1: Representación de un proceso Gaussiano con función de media $m(\mathbf{x})$ (línea azul) y función de covarianzas $k(\mathbf{x}, \mathbf{x}') = 2I$ (su diagonal, franja azul claro, representa el intervalo de 95 % de confianza). Se ha dibujado en negro la distribución Gaussiana del punto $x = 6$, puede verse que también tiene forma Gaussiana. [Elaboración propia]

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \tag{2.1}$$

donde μ es la media de la distribución y σ^2 es su varianza, ambos escalares. Si se sustituyesen por funciones, se tendría una idea bastante cercana y suficiente sobre qué son los procesos Gaussianos. Se puede pensar en estas funciones como vectores infinitos, por lo que puede surgir la duda de si su tratamiento dará problemas. Sin embargo, en la práctica se trabaja con un conjunto finito de puntos, mediante los cuales es posible aplicar inferencia utilizando un proceso Gaussiano para así obtener una distribución predictiva del resto de valores.

La expresión de la distribución Gaussiana multivariante está enteramente definida por su vector de medias $\boldsymbol{\mu}$ y su matriz de covarianzas $\boldsymbol{\Sigma}$ por la siguiente ecuación

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}\sqrt{|\boldsymbol{\Sigma}|}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right\} \tag{2.2}$$

donde D corresponde al número de dimensiones de la distribución. A fin de realizar una representación de esta distribución se pierde información, ya que no es posible representar infinitas dimensiones. Es por ello que para esta representación solo se utiliza el vector de medias $\boldsymbol{\mu}$ y la matriz de covarianzas $\boldsymbol{\Sigma}$, como puede verse en la figura 2.1. En esta, cada uno de los puntos con $f(x) = 0$ corresponde a una distribución Gaussiana unidimensional con $\mu = 0$ y $\sigma = 2$. Una propiedad importante de la distribución Gaussiana multivariante es que si dos conjuntos de variables son conjuntamente Gaussianas, entonces su distribución conjunta también será Gaussiana. De forma inversa, si $(\mathbf{x}_1, \mathbf{x}_2) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, donde

$$\begin{pmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{pmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12}\\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{bmatrix}\right)$$

Entonces, $\mathbf{x}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11})$ [1, Sección 2.3]. Esta propiedad, que será de utilidad más adelante, surge del requerimiento de consistencia de la definición de los procesos Gaussianos.

2.2. Procesos gaussianos en regresión

En la sección 2.1, se explicó un idea intuitiva sobre qué son los procesos Gaussianos y se representó un proceso Gaussiano con función de media $m(\mathbf{x}) = 0$ y función de covarianzas $k(\mathbf{x}, \mathbf{x}') = 2I$. En esta sección, se hará un planteamiento de los procesos Gaussianos en regresión siguiendo [2, Capítulo 2]. Así pues, se dará la definición formal de procesos Gaussianos y se trabajará con ellos, para que sea posible aplicarlos en un problema de regresión, donde el objetivo de realizar predicciones.

Definición 2.1 *Un proceso Gaussiano es una distribución de probabilidad sobre funciones, de tal manera que al escoger un conjunto finito de puntos y evaluar la distribución de probabilidad de la función en esos puntos, la distribución conjunta es una Gaussiana multivariante.*

Un proceso Gaussiano está completamente definido por su media $m(\mathbf{x})$ y su covarianza $k(\mathbf{x}, \mathbf{x}')$, donde

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{2.3a}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \tag{2.3b}$$

Así que, este se representará como

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{2.4}$$

A continuación, se aplicará un punto de vista Bayesiano a la regresión lineal, al igual que [2, Capítulo 2] [1, Sección 6.4], a fin de mostrar como la representación mediante funciones surge de manera natural al partir desde el punto de vista de los vectores. En la sección 2.1 se habló de la falta de expresividad que tenían las funciones lineales en el problema de regresión. Una posible solución a este problema sería realizar una transformación de las entradas por medio de funciones base $\phi(\cdot)$, a fin de pasar el problema a un espacio de mayor dimensionalidad, donde si fuese resoluble linealmente.

De forma que la $función\ subyacente\ f(\mathbf{x})$ que define un problema de regresión vendría dada por la siguiente ecuación

$$f(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\phi(\mathbf{x}) \tag{2.5}$$

Suponiendo que los pesos $\mathbf{w}$ están definidos por una Gaussiana isotrópica de media un vector de ceros $\mathbf{0}$ y matriz de covarianzas $\mathbf{\Sigma}_p$

$$p(\mathbf{w}) \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_p) \tag{2.6}$$

Como se quiere obtener la distribución de probabilidad de $f(\mathbf{x})$ que estará definida enteramente por su media $\mathbb{E}[f(\mathbf{x})]$ y su covarianza $\mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$, considerando las ecuaciones 2.5 y 2.6 se puede obtener

$$\mathbb{E}[f(\mathbf{x})] = \boldsymbol{\phi}(\mathbf{x})^{\mathsf{T}}\mathbb{E}[\mathbf{w}] = \mathbf{0}$$
$$\mathbb{E}[(f(\mathbf{x})(f(\mathbf{x}'))] = \boldsymbol{\phi}(\mathbf{x})^{\mathsf{T}}\mathbb{E}[\mathbf{w}\mathbf{w}^{\mathsf{T}}]\boldsymbol{\phi}(\mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^{\mathsf{T}}\mathbf{\Sigma}_p\boldsymbol{\phi}(\mathbf{x}')$$

Teniendo en cuenta que $\mathbf{\Sigma}_p$ es una matriz positiva definida se puede calcular $\mathbf{\Sigma}_p^{1/2}$, utilizando la descomposición LDL donde D es una matriz diagonal, en la que se cumple que si $A = LDL$, entonces $A^{1/2}= LD^{1/2}L$. Por lo tanto, si se utiliza $\psi(\mathbf{x}) = \mathbf{\Sigma}_p^{1/2}\phi(\mathbf{x})$, se puede expresar $K(\mathbf{x}, \mathbf{x}') = \psi(\mathbf{x})^{\mathsf{T}}\psi(\mathbf{x}')$. Además, este cambio resulta muy conveniente, puesto que, si un algoritmo es expresado en términos del producto interno de su espacio de entradas, se puede definir ese algoritmo directamente sobre el espacio de características sustituyendo dichos productos internos. Este cambio se conoce como $truco\ del\ kernel$ o $sustitución\ del\ kernel$, y permite hacer el siguiente cambio

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}') = \psi(\mathbf{x})^{\mathsf{T}}\psi(\mathbf{x}') &= \begin{bmatrix} \psi(\mathbf{x}_1)^{\mathsf{T}}\psi(\mathbf{x}_1) & \psi(\mathbf{x}_1)^{\mathsf{T}}\psi(\mathbf{x}_2) & \dots & \psi(\mathbf{x}_1)^{\mathsf{T}}\psi(\mathbf{x}_n) \\ \psi(\mathbf{x}_2)^{\mathsf{T}}\psi(\mathbf{x}_1) & \psi(\mathbf{x}_2)^{\mathsf{T}}\psi(\mathbf{x}_2) & \dots & \psi(\mathbf{x}_2)^{\mathsf{T}}\psi(\mathbf{x}_n) \\ \dots & \dots & \dots & \dots \\ \psi(\mathbf{x}_n)^{\mathsf{T}}\psi(\mathbf{x}_1) & \psi(\mathbf{x}_n)^{\mathsf{T}}\psi(\mathbf{x}_2) & \dots & \psi(\mathbf{x}_n)^{\mathsf{T}}\psi(\mathbf{x}_n) \end{bmatrix} \\ &= \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \dots & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \dots & K(\mathbf{x}_2, \mathbf{x}_n) \\ \dots & \dots & \dots & \dots \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & \dots & K(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \end{aligned} \tag{2.7}$$

Esta matriz K es la matriz de Gram o la matriz kernel. Al realizar este cambio la complejidad de calcular K está en función del número de datos que se tenga, en lugar del número atributos en el espacio extendido (que podrían ser infinitos).

2.2.1. Distribución predictiva

Al final de la sección 2.2 se sustituyó en la ecuación 2.6 los términos de las funciones base $\phi(\mathbf{x})$ y de la matriz de pesos $\mathbf{\Sigma_p}$, con el objetivo de utilizar directamente las entradas en la nueva función de covarianzas definida por el kernel. Esta será la expresión del priori, que viene dada por

$$p(\mathbf{f}) \sim \mathcal{N}(m(\cdot)), k(\cdot,\cdot)) \tag{2.8}$$

donde $\mathbf{f} = \mathbf{f}(X)$. Si se utiliza esta expresión para predicción, se estaría calculando la distribución predictiva del priori, (representado en las figuras 2.1 y 2.2(a)). En esta última figura se muestrearon diez funciones. Su expresión matemática es la siguiente

$$p(\mathbf{f}_*|X_*) \sim \mathcal{N}(\mathbf{0}, K_*(X_*, X_*)) \tag{2.9}$$

donde $\mathbf{f}_* = \mathbf{f}(X_*)$ y X_* son los puntos donde se está prediciendo, aunque al no haber datos las predicciones solo muestran la naturaleza del priori. Con el propósito de utilizar los datos observados para así poder predecir otros nuevos, se tiene que calcular la distribución predictiva a posteriori. Esta distribución utiliza las observaciones condicionadas a los datos, para restringir los posibles lugares por los que pueden pasar las funciones. Se puede fijar una variable de error, a fin de tener en cuenta el posible ruido, que se hará al final de este apartado.

Se supuso que la distribución a priori de los datos en la ecuación 2.8 y las predicciones en la ecuación 2.9 eran Gaussianas, por lo tanto, la distribución conjunta también será Gaussiana. En consecuencia, es posible aplicar los resultados de [1, Sección 2.3] para calcularla. De modo que al resolver

$$p(\mathbf{f}, \mathbf{f}_*|X, X_*) = \begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} |X, X_* \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right) \tag{2.10}$$

se obtiene la ecuación de la distribución predictiva.

$$p(\mathbf{f}_*|\mathbf{x}_*, X, \mathbf{y}) \sim \mathcal{N}(K(X_*,X)K(X,X)^{-1}\mathbf{f}, \\ K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*))$$

Con la intención de simplificar su forma, se realizarán los siguientes cambios en la notación: $K_*^{\mathsf{T}} = K(X_*X)$, $K^{-1} = K(X,X)^{-1}$ y $K_* = K(X,X_*)$. De forma que la expresión de la distribución predictiva $p(\mathbf{f}_*|\mathbf{x}_*, X, \mathbf{y})$ quedaría

$$p(\mathbf{f}_*|\mathbf{x}_*, X, \mathbf{y}) \sim \mathcal{N}(K_*^{\mathsf{T}}K^{-1}\mathbf{y}, K_{**} - K_*^{\mathsf{T}}K^{-1}K_*) \tag{2.11}$$

En la figura 2.2(b) se representa el resultado de utilizar esta expresión, en la cual se muestrearon

diez funciones. Sin embargo, son pocos los problemas que no presentan ruido ε. Teniendo esto en cuenta se va a añadir una constante de ruido definida por

$$\varepsilon \sim \mathcal{N}(0, \sigma_n^2 I) \tag{2.12}$$

en la ecuación de la función predictiva. De forma que ahora se definirán las observaciones en función de $\mathbf{y}$, que viene dado por

$$\mathbf{y} = \mathbf{f} + \varepsilon$$

Al hacer este cambio, se está sumando σ_n^2 a los elementos de la diagonal de la función de covarianzas, de tal modo que la distribución conjunta de las observaciones, ahora ruidosas junto con los datos quedaría como

$$p(\mathbf{y}, \mathbf{f}_*|X, X_*) = \begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} |X, X_* \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K & K_* \\ K_*^\mathsf{T} & K_{**} \end{bmatrix}\right) \tag{2.13}$$

Por lo tanto, las funciones media y varianza que se utilizarán para predecir son las siguientes

$$\bar{f}_* = K_*^\mathsf{T}(K + \sigma_n^2 I)^{-1}\mathbf{y} \tag{2.14a}$$

$$\mathbb{V}[f_*] = K_{**} - K_*^\mathsf{T}(K + \sigma_n^2 I)^{-1}K_*) \tag{2.14b}$$

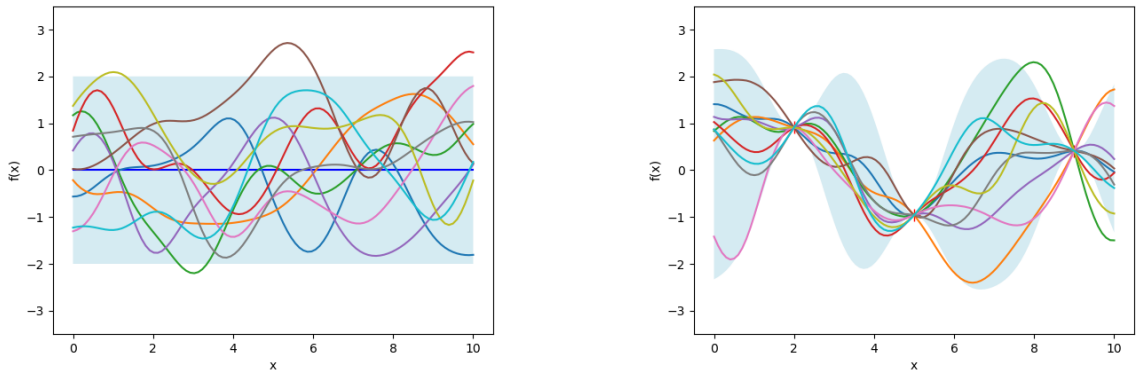
Nótese que $\mathbf{f}_*$ no ha sido modificado. También es posible añadir una componente de ruido a las predicciones efectuando el procedimiento anterior, de forma que se pasaría a predecir $\mathbf{y}_*$. Sin embargo, las predicciones que se han realizado en este trabajo han sido obtenidas sin realizar este cambio, ya que el objetivo es predecir $\mathbf{f}_*$ en lugar de $\mathbf{y}_*$.

2.2.2. Verosimilitud marginal

Cuando se introdujo la función kernel como función de covarianzas, se introdujeron junto con él los hiperparámetros que lo regulan. El tema de la optimización de los hiperparámetros será discutido en la sección 2.4. Sin embargo, será útil introducir ahora la verosimilitud marginal, puesto que se necesitará en dicha sección, a fin de obtenerla se deben marginalizar las evidencias respecto del priori de la siguiente forma

$$p(\mathbf{y}|X) \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|X)d\mathbf{f}$$

Sabiendo que la verosimilitud también es una distribución Gaussiana, su expresión viene dada por



(a) Priori (b) Posteriori

Figura 2.2: En ambas figuras se muestran diez funciones, las cuales han sido generadas mediante la distribución predictiva del priori en 2.2(a) y del posteriori (después de realizar tres observaciones) en 2.2(b). En ambas figuras el kernel utilizado como función de covarianza fue el $squared - exponential$. El $length - scale$ fue fijado a 1; la franja azul corresponde al intervalo de 95 % de confianza y es de dos desviaciones estándar. [Elaboración propia]

$$\begin{aligned} p(\mathbf{y}|X) &\sim \mathcal{N}(\mathbf{0}, K + \sigma_n^2 I) \\ &= \frac{1}{(2\pi)^{n/2}|K + \sigma_n^2 I|^{1/2}} \exp\left(\frac{1}{2}\mathbf{y}^\mathsf{T}(K + \sigma_n^2 I)^{-1}\mathbf{y}\right) \end{aligned} \tag{2.15}$$

El coste computacional de calcular esta ecuación vendrá determinado por la inversión de la matriz $(K + \sigma_n^2 I)^{-1}$. Realizar esta inversión de matrices es una operación costosa computacionalmente, puesto que es de orden $O(n^3)$. Con el propósito de resolverla se suele utilizar la descomposición de Cholesky, que es más estable y algo más rápida que el método tradicional, pero sigue teniendo el mismo orden de complejidad. De hecho, la librería que se ha utilizado en el desarrollo de la herramienta, tratada en la sección 4.1, utiliza este algoritmo en lugar de invertir la matriz directamente.

2.3. Kernels

Una de las aplicaciones de los kernels son los procesos Gaussianos. Se puede obtener una idea intuitiva sobre el funcionamiento de los kernels, si se piensa que son equivalentes a tener infinitas funciones base que transformen el espacio de entradas en otro de mayor dimensionalidad. Una función se denomina kernel cuando cumple los requisitos de ser simétrica y positiva semidefinida, y se utiliza para medir la similitud entre dos puntos.

Un tipo de kernels son los $estacionarios$, los cuales solo dependen de $\mathbf{x} - \mathbf{x}'$, y se conocen así por ser invariantes a traslaciones en el espacio de entradas. Otro tipo son los $homogéneos$, también conocidos como $isotrópicos$ o $radial\ basis\ function$, que solo dependen de la magnitud de la distancia de los puntos de entrada $||\mathbf{x} - \mathbf{x}'||$ [2, Capítulo 4], [1, Capítulo 6].

Existen ecuaciones que se pueden utilizar para construir kernels, que se deben usar si se quiere que tengan ciertas propiedades de interés y que este sea válido. En [1, Capítulo 6.2] están recogidas estas ecuaciones. Una alternativa a este procedimiento sería demostrar que una determinada función, efectivamente cumple las propiedades de un kernel, sin embargo, esto no siempre será sencillo de llevar a cabo.

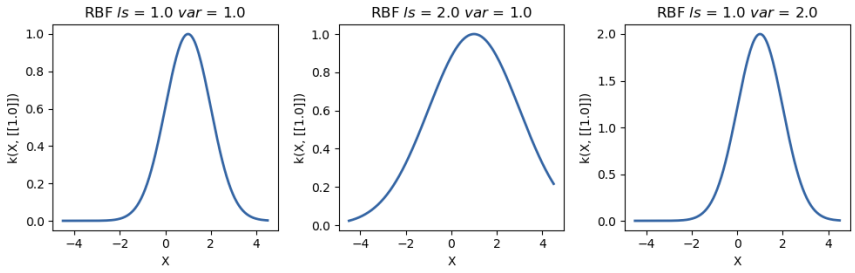


Figura 2.3: Representación del kernel $squared - exponential$ o RBF. Se han variado los hiperparámetros de $length - scale$ y $varianza$ para ver sus efectos. En los títulos de las figuras están especificados los valores de los hiperparámetros. Se puede observar cómo al incrementar ls aumenta la anchura, mientras que al incrementar la varianza aumenta su altura. Nótese que la tercera figura es similar a la primera, sin embargo, los valores del eje de ordenadas están duplicados. [Elaboración propia]

En la parte experimental de este trabajo se han utilizado dos kernels: SE y Matérn52. El primero de ellos ya fue utilizado en el proceso Gaussiano de las figuras 2.2(a) y 2.2(b), correspondiendo a $squared - exponential$ (SE), que ha sido ampliamente utilizado en la literatura de los procesos Gaussianos. Este kernel también es conocido como RBF ($radial\ basis\ function$). Su forma se presenta en la figura 2.3 para diferentes valores de $lenght - scale$ y $varianza$. Su expresión matemática viene dada por

$$k_{SE}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(\frac{1}{2l^2}||\mathbf{x}_i - \mathbf{x}_j||^2\right) \tag{2.16}$$

donde σ_f es la varianza de la función, y l es el $length - scale$ (se encarga de regular cuanto varia la función entre dos puntos).

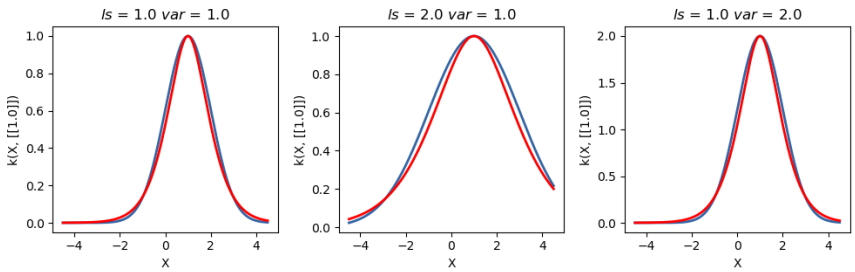


Figura 2.4: Representación del kernel SE (azul) y el de Matérn52 con $\nu = 5/2$ (rojo) para diferentes valores de $length - scale$ y de $varianza$ (especificados en los títulos las figuras). [Elaboración propia]

Los efectos de variar estos hiperparámetros de los kernels en los procesos Gaussianos serán explicados en la sección 2.3.1, y las funciones generadas por el kernel RBF se pueden ver en 2.6(a) para el priori y en 2.6(b) para el posteriori. El otro kernel que será utilizado es el kernel de Matérn [3]. Este kernel dispone de un parámetro ν, el cual permite que se regule la flexibilidad del kernel. Su expresión viene dada por

$$k_{Matérn}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2^{\nu-1}\Gamma(\nu)}(\sqrt{2\nu}||\mathbf{x}_i - \mathbf{x}_j||)^{\nu} H_{\nu}(\sqrt{2\nu})||\mathbf{x}_i - \mathbf{x}_j|| \tag{2.17}$$

donde $\Gamma(\cdot)$ es la función Gamma y $H(\cdot)$ la función de Bessel. Si $\nu \to \infty$, entonces el kernel de Matérn coincide con el kernel SE. Este ha sido utilizado fijando $\nu = 5/2$, tras lo cual la ecuación queda de la siguiente forma

$$k_{Matérn52}(r) = \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5}r}{l}\right) \tag{2.18}$$

donde $r = ||\mathbf{x}_i - \mathbf{x}_j||$. En cada gráfica de la figura 2.4 se muestran los dos kernels, con los mismos hiperparámetros que los especificados en la figura 2.3. Se puede apreciar como el kernel Matérn52 es un poco más estrecho que SE, lo que producirá que dos puntos separados entre sí estén menos correlacionados al utilizar este kernel que al utilizar SE. Este efecto provocará que el kernel Matérn52 suponga que la función es menos suave que la que supone SE, esto puede verse en la figura 2.5, donde se muestra una comparativa entre la distribución predictiva a posteriori de un proceso Gaussiano con kernel SE (o RBF) y otro con kernel de Matérn52. En la sección 2.4 se verán los efectos de variar los hiperparámetros de los kernels.

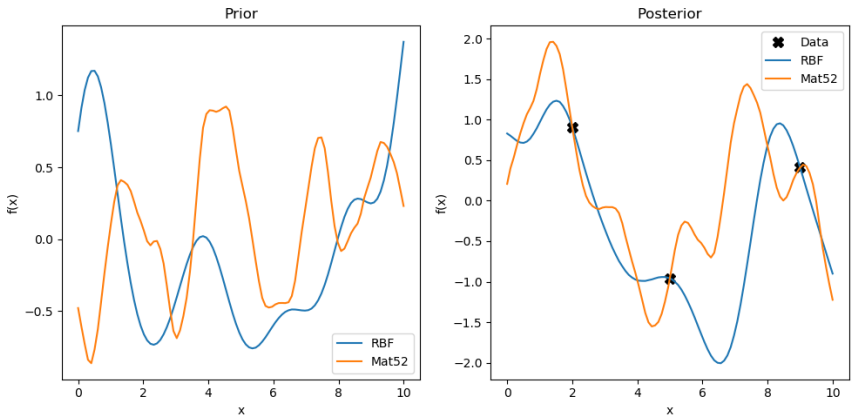
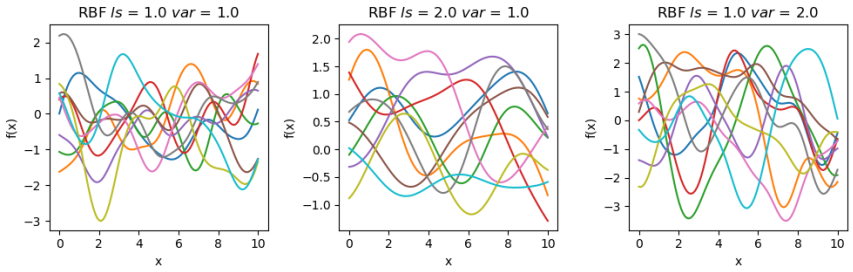


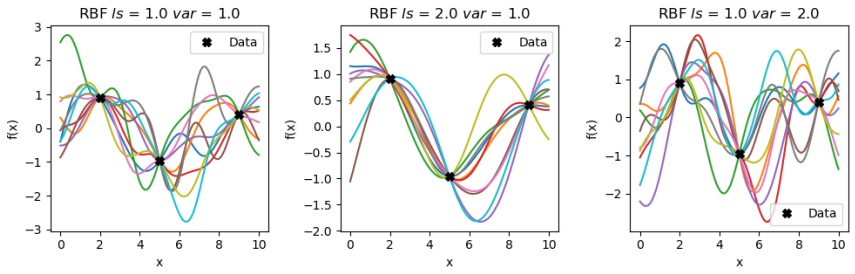
Figura 2.5: Representación comparativa entre las distribuciones a priori y posteriori para los kernels RBF y Matérn52. [Elaboración propia]

2.3.1. Hiperparámetros del kernel

Se ha estado hablando sobre kernels, sin embargo, se ha estado pasando por alto como afectan a su forma las diferentes configuraciones de hiperparámetros que pueden tener. Estos efectos pueden verse en las gráficas de las figuras 2.6(a) y 2.6(b), donde se han muestreado diez funciones en cada gráfica. Las funciones presentadas en las gráficas de la figura 2.6(a) han sido obtenidas del priori, mientras que las de 2.6(b) han sido obtenidas del posteriori. Se puede apreciar como en los modelos cuyo kernel tiene un menor valor de $length - scale$, presentan una función más ondulada. Además, es posible ver que la varianza regula la amplitud. También es llamativo que las funciones obtenidas mediante el posteriori con un $ls = 2{,}0$ parecen ajustarse más a la función subyacente.



(a) Funciones kernel generadas con SE del priori



(b) Funciones kernel generadas con SE del posteriori

Figura 2.6: Representación de tres procesos Gaussianos kernel RBF pero con diferentes hiperparámetros. En la figura 2.6(a) se obtuvieron del priori y en la 2.6(b) se obtuvieron del posteriori (con tres observaciones). [Elaboración propia]

2.4. Optimización de los hiperparámetros

Una de las buenas características de los procesos Gaussianos es que proveen de una expresión para la verosimilitud marginal que es analíticamente tratable, en contraste con otros modelos del aprendizaje automático que no cuentan con dicha expresión. Esta ecuación es interesante porque se puede utilizar para optimizar los hiperparámetros del kernel, los cuales se habían estado pasando por alto hasta ahora. Esta optimización se puede llevar a cabo maximizando la verosimilitud marginal dada por

la ecuación 2.15, para lo cual en primer lugar se aplicará el logaritmo para que sea más manejable, de forma que quedaría

$$p(\mathbf{y}|X,\boldsymbol{\theta}) = -\frac{1}{2}(\mathbf{y}^{\mathsf{T}}(K+\sigma_n^2)^{-1}\mathbf{y}) - \frac{1}{2}\log|K+\sigma_n^2 I| - \frac{n}{2}\log 2\pi \tag{2.19}$$

donde $\boldsymbol{\theta}$ son los hiperparámetros del kernel que se utiliza en el proceso Gaussiano. Los tres términos de la parte derecha de la igualdad significan lo siguiente (comenzando desde la izquierda): el primero está relacionado con como de bien se ajusta el modelo (la función que describe el proceso Gaussiano) a los datos; el segundo mide la complejidad del modelo, de forma que cuanto menor sea el valor del determinante, menor será la penalización de este término; finalmente, el tercer término es una función lineal que tiene en cuenta el número de datos y se encarga de normalizar.

Continuando con la optimización, será necesario calcular el gradiente de 2.19 para su maximización. Al seguir el desarrollo de [2, Sección 5.4] se obtiene

$$\begin{aligned}\frac{\partial}{\partial\theta_j}p(\mathbf{y}|X,\boldsymbol{\theta}) &= \frac{1}{2}\left(\mathbf{y}^{\mathsf{T}}K^{-1}\frac{\partial K}{\partial\theta_j}K^{-1}\mathbf{y}\right) - \frac{1}{2}\mathrm{tr}\left(K^{-1}\frac{\partial K}{\partial\theta_j}\right)\\ &= \frac{1}{2}\mathrm{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathsf{T}}K^{-1})\frac{\partial K}{\partial\theta_j}\right)\end{aligned} \tag{2.20}$$

donde $\boldsymbol{\alpha} = (K^{-1}\mathbf{y})$. El coste de calcular el gradiente viene determinado por la inversión de la matriz K^{-1}, que es $O(n^3)$. Sin embargo, una vez que ha sido obtenida esta inversa, el coste del gradiente viene determinado por el coste de calcular las derivadas de los hiperparámetros $\partial K/\partial\theta_j$, que es $O(n^2)$. Como suele haber varios máximos, se suelen utilizar métodos de ascenso quasi-Newton con múltiples puntos distintos de partida [4].

Este método de obtener los hiperparámetros tiende a hacer $overfitting$ cuando hay pocos datos. Debido a que, si el kernel es muy expresivo, se ajustan sus parámetros para maximizar la probabilidad de los datos ya observados.

3

OPTIMIZACION BAYESIANA

La optimización Bayesiana, como su nombre indica, es una que técnica se utiliza para optimizar una $función\ objetivo\ f(\mathbf{x})$ (también llamada $latente$ o $subyacente$). Su aplicación se hace en escenarios donde las observaciones tienen un coste elevado, pueden tener ruido y no se tiene una expresión para $f(\mathbf{x})$. Dadas estas tres características, el objetivo es obtener los valores que además de dar la mayor cantidad de información, minimicen la $función\ objetivo$ en el menor número posible de observaciones. Escenarios como el anterior descrito podrían darse en muy diferentes contextos, por ejemplo entrenar una inteligencia artificial para que controle un taxi [5], donde cada evaluación significaría entrenar y probar una nueva red, o sintetizar una molécula con unas características determinadas, donde cada evaluación puede requerir realizar un costoso experimento real en tiempo y en dinero. En ambos casos se quiere obtener la configuración de parámetros, dada por el vector $\mathbf{x}_*$, que obtenga el error más pequeño en $f(\mathbf{x})$. Por lo tanto, la expresión matemática que recoge esto sería la siguiente

$$\mathbf{x}_* = \min_{\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d} f(\mathbf{x}) \tag{3.1}$$

donde $\mathbf{x}_*$ es el valor de entrada, el cual minimiza la $función\ subyacente\ f(\mathbf{x})$, y $\mathcal{X}$ es el espacio de características donde se está optimizando. Nótese que si se elige mal $\mathcal{X}$ no se optimizará correctamente $f(\mathbf{x})$. Con el propósito de efectuar esta minimización, se llevará a cabo un $trade-off$ entre explotar soluciones prometedoras y explorar zonas desconocidas del espacio de entradas. Para conseguir este $trade-off$ la función de adquisición utilizará la media $\mu(\mathbf{x}_{n+1})$ y la covarianza $\sigma^2(\mathbf{x}_{n+1})$ dadas por las ecuaciones 2.14a y 2.14b, respectivamente, obtenidas del modelo creado por el proceso Gaussiano ($\mathcal{GP}$), en base a las evidencias que se tienen de la $función\ objetivo$. Con el propósito de calcular así la utilidad esperada de realizar una observación en un determinado punto $\mathbf{x}_{n+1}$. Por lo tanto, la optimización Bayesiana es una técnica que realiza sus predicciones en base a la creencia que tiene sobre el modelo. Este tipo de aproximación a los problemas obtiene mejores resultados que realizar una simple búsqueda en rejilla o una búsqueda aleatoria, puesto que ninguna de estas dos estrategias utiliza el modelo para guiar el proceso de minimización. En el capítulo 5 se mostrará una comparativa de los resultados obtenidos utilizando una búsqueda aleatoria y optimización Bayesiana.

Un ejemplo de la aplicación de la optimización Bayesiana en un problema de optimización de 1 dimensión se ilustra en la figura 3.1. En la columna izquierda se ilustra la distribución predictiva del

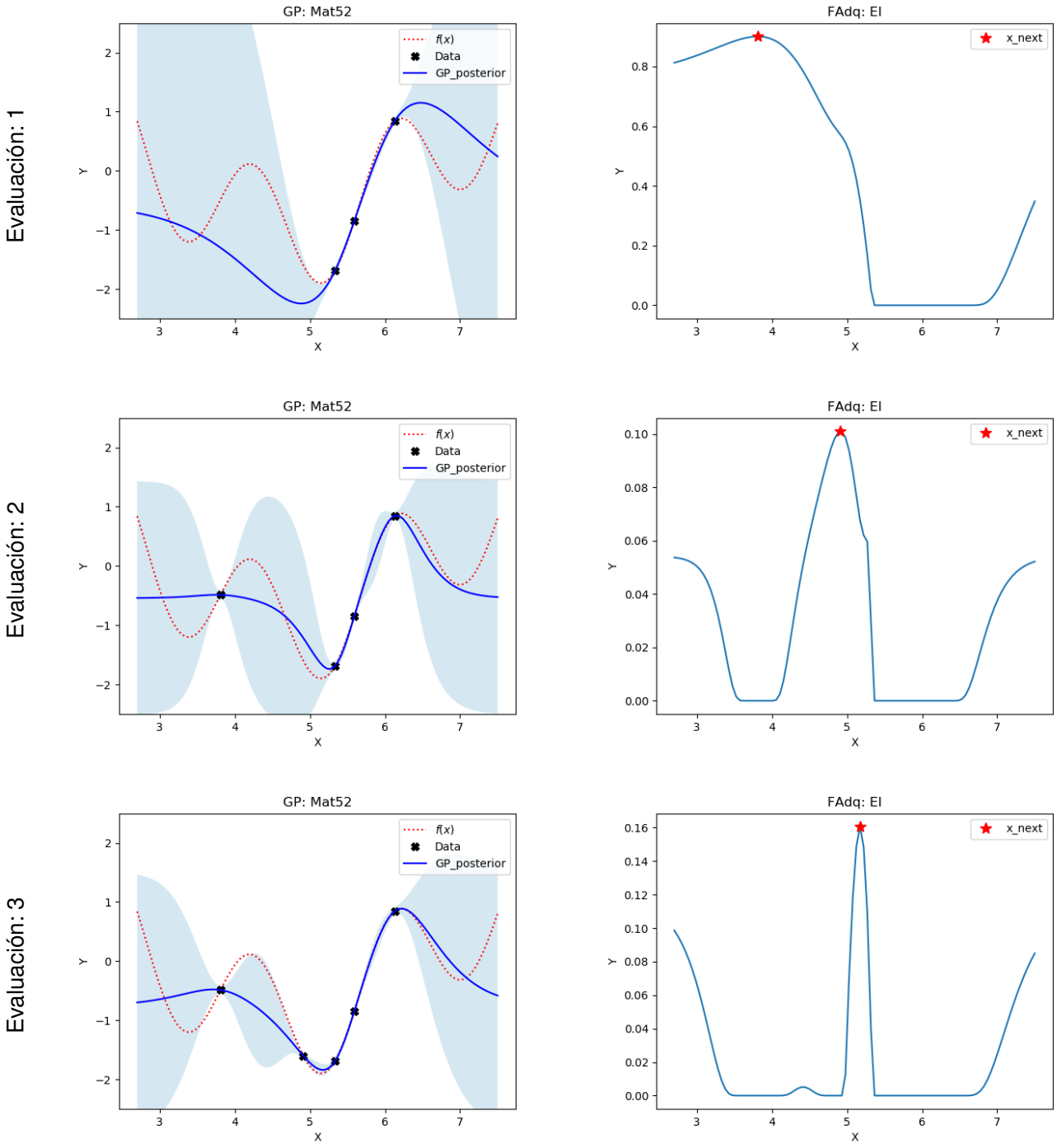


Figura 3.1: La columna de la izquierda corresponde a imágenes de la distribución predictiva a posteriori de un $\mathcal{GP}$ (en azul la media y azul claro sombreado su covarianza), junto a la $función\ objetivo$ $f(\mathbf{x})$ (línea punteada en rojo). En columna de la derecha se representa la función de adquisición de la optimización Bayesiana. [Elaboración propia]

input : l número de evaluaciones a realizar
for $i = 1, 2, ..., l$ **do**
    actualizar el $\mathcal{GP}$;
    calcular $\alpha(\cdot)$ obtener $\mathbf{x}_{n+1}$ mediante la optimización de $\alpha(\cdot)$: $\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x}; \mathcal{D}_n)$;
    medir $y_{n+1} = f(\mathbf{x}_{n+1}) + \varepsilon$;
    guardar y_{n+1} en $\mathcal{D}_n$: $\mathcal{D}_n = \{\mathcal{D}_n, \mathbf{x}_{n+1}, y_{n+1}\}$;
end
recomendar $\mathbf{x}_{best}$, que corresponde al mejor valor encontrado

Algoritmo 3.1: Algoritmo que ilustra el proceso que se sigue en la optimización Bayesiana.

$\mathcal{GP}$, mediante su media y su varianza. En la columna derecha se representa la utilidad en cada punto dada por la función de adquisición. El proceso de minimización de la $función\ subyacente$ comenzó con tres datos, elegidos aleatoriamente, y a partir de ahí se fue aplicando el algoritmo de optimización 3.1. Se puede apreciar cómo en cada iteración tras actualizar el $\mathcal{GP}$, se evalúa la utilidad de medir en cada punto, mediante la función de adquisición $\alpha(\cdot)$, y es en ese nuevo punto donde se realiza la siguiente evaluación. El kernel utilizado por el $\mathcal{GP}$ fue Matérn52, y la función de adquisición fue $expected\ improvement$ (EI), que se describirá en la sección 3.1.

Por lo tanto, la idea detrás de la optimización Bayesiana consiste en utilizar el modelo por medio del $\mathcal{GP}$, para así calcular el siguiente mejor punto a evaluar, el cual se calcula utilizando la función de adquisición, encargada de medir la utilidad en cada punto. De esta forma, el problema pasa a ser optimizar la función de adquisición en cada evaluación, cuyo coste se considera despreciable en comparación con evaluar en la $función\ objetivo$, carece de ruido y se cuenta con su forma explícita, por lo que es más fácil de optimizar.

3.1. Función de adquisición

La función de adquisición es de gran importancia dentro de la optimización Bayesiana, puesto que regula el $trade - off$ entre explotación y exploración. Existen múltiples métodos utilizados, como pueden ser $probability\ of\ improvement$ (PI), $expected\ improvement$ (EI), $lower\ confidence\ bound$ (LCB), $entropy\ search$ (ES), y un portafolios de varias estrategias, esta última suele dar mejores resultados [4]. Los valores devueltos por la función de adquisición suelen corresponder con los valores que se espera tengan una mayor mejora en la tarea de optimización [5].

La primera de las estrategias enumeradas, $probability\ of\ improvement$, fue propuesta en 1964 por Kushner en [6]. Como su nombre indica, mide la probabilidad de que la próxima observación sea mejor que el mejor valor obtenido hasta el momento. Como los valores de $f(\mathbf{x})$ son los de una distribución posterior Gaussiana, la expresión matemática que define la probabilidad de mejora viene dada por

$$\begin{aligned}\mathbf{PI}(\mathbf{x}) &= P(f(\mathbf{x}) \leq f(\mathbf{x}^+) - \xi) \\ &= \Phi\left(\frac{f(\mathbf{x}^+) + \xi - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right)\end{aligned} \tag{3.2}$$

donde $\Phi(\cdot)$ es la función de probabilidad acumulada de una distribución Gaussiana estándar, ξ es una constante de regularización y $f(\mathbf{x}^+)$ es el mejor valor obtenido hasta ese momento. El problema de esta estrategia es que es pura explotación [5], es decir, una vez que hay un punto con un buen resultado, dirige la búsqueda únicamente en las cercanías a dicho punto por lo que tiende a quedarse en mínimos locales.

Con el objetivo de tener en cuenta también las zonas no exploradas, se tiene que cambiar el enfoque para maximizar $la\ mejora\ esperada$. Con la finalidad de lograr esto primero se debe definir qué es la mejora, de forma que siguiendo [5], se obtiene

$$\mathbf{I}(\mathbf{x}) = \text{máx}\{0, f(\mathbf{x}^+) - f(\mathbf{x}_{n+1})\} \tag{3.3}$$

En la cual $\mathbf{I}(\mathbf{x})$ solo tendrá valores positivos cuando la evaluación en el nuevo punto $\mathbf{x}_{n+1}$ sea menor que el anterior valor más bajo $f(\mathbf{x}^+)$. A fin de continuar con el cálculo de la mejora esperada, se debe tener en cuenta que los valores de $f(\mathbf{x})$ estaban definidos por una Gaussiana, por lo que la probabilidad de mejorar $\mathbf{I}(\cdot)$ en la distribución posterior definida por $\mu(\mathbf{x})$ de la ecuación 2.14a y $\sigma^2(\mathbf{x})$ de la ecuación 2.14b, se obtiene al resolver la siguiente integral

$$\mathbb{E}(\mathbf{I}) = \int_0^\infty \mathbf{I} \frac{1}{\sqrt{2\pi}\sigma(\mathbf{x})} \exp\left(-\frac{f(\mathbf{x}^+) + \xi - \mu(\mathbf{x}) - \mathbf{I})^2}{2\sigma^2(\mathbf{x})}\right) d\mathbf{I}$$

donde $\mathbf{I} = \mathbf{I}(\mathbf{x})$, da como resultado la mejora esperada

$$\mathbf{EI}(\mathbf{x}) = \begin{cases} (f(\mathbf{x}^+) + \xi - \mu(\mathbf{x}))\Phi(Z) + \sigma(\mathbf{x})\phi(Z) & \text{si } \sigma(\mathbf{x}) > 0 \\ 0 & \text{si } \sigma(\mathbf{x}) = 0 \end{cases} \tag{3.4}$$

donde

$$Z = \frac{f(\mathbf{x}^+) + \xi - \mu(\mathbf{x})}{\sigma(\mathbf{x})}$$

donde $\mu(\cdot)$ es la función de densidad de probabilidad de una distribución Gaussiana estándar.

Tanto en 3.2 como en 3.4 la constante ξ de regulación, que es sumada a $f(\mathbf{x}^+)$, se utiliza para especificar que podría haber un valor superior al mejor encontrado.

Por último, se hablará de la función de adquisición que utiliza el límite de menor confianza (o el de mayor, si se está maximizando). Esta función es optimista en cuanto a la varianza $\sigma(\mathbf{x})$. A continuación, se presenta la expresión matemática de esta estrategia

$$\mathbf{LCB}(\mathbf{x}) = \mu(\mathbf{x}) - \nu\sigma(\mathbf{x}) \tag{3.5}$$

donde ν es una constante llamada $kappa$. Nótese que se debe cumplir que $\nu \geq 0$ (puesto que se quiere minimizar). Una representación del tipo de elecciones para el $\mathbf{x}_{n+1}$ que proveen estas tres funciones de adquisición, para diferentes valores de las constantes reguladoras de cada una de ellas, se observa en la figura 3.2 (tomada de [5]). Aunque en dicho artículo el objetivo es maximizar, es equivalente al objetivo de minimización.

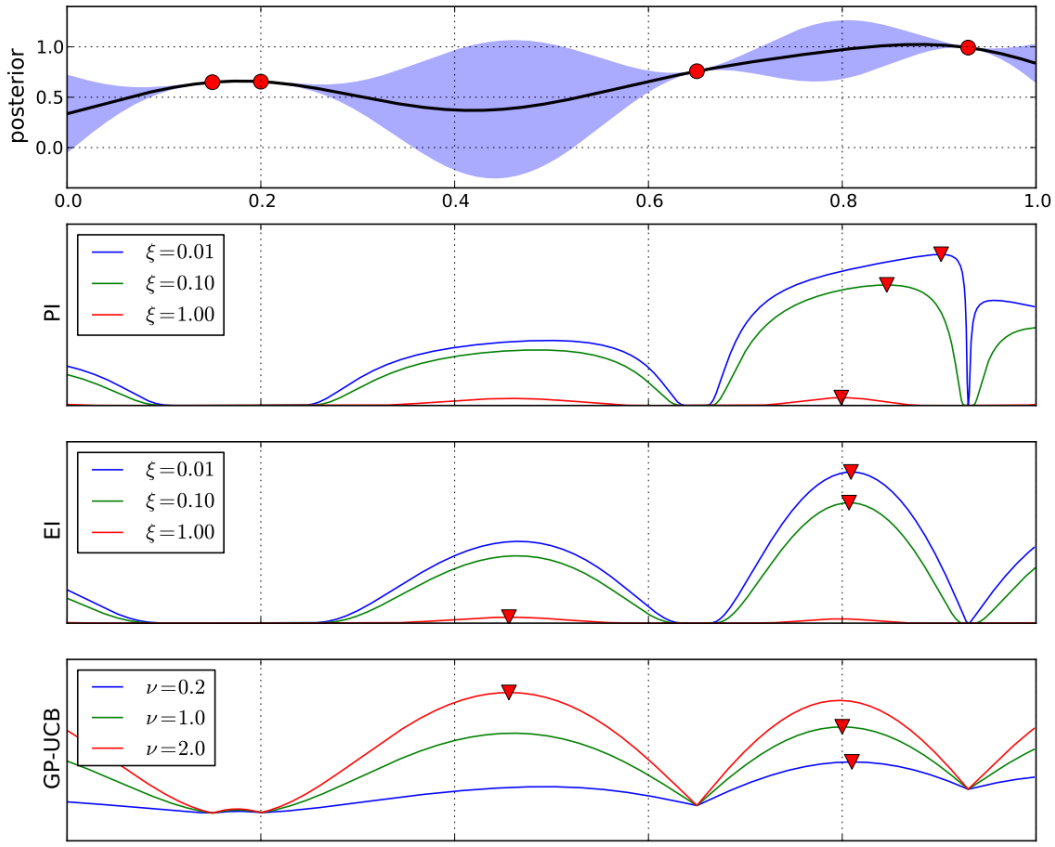


Figura 3.2: La primera imagen corresponde a la distribución posteriori definida por $\mu(\mathbf{x})$ y $\sigma(\mathbf{x})$. Las otras tres imágenes son una comparativa de los procesos de maximización de la $función\ subyacente$, por medio de las funciones de adquisición PI, EI y GP-UCP (su equivalente en minimización es el GP-LCB que ha sido tratado en esta sección), cada una de ellas para diferentes valores de los hiperparámetros. [Imagen obtenida de [5]]

Finalmente, se debe añadir que para la maximización de la función de adquisición existen diferentes estrategias: partir el espacio en varias zonas, realizar una búsqueda en rejilla adaptativa o utilizar múltiples puntos desde los que realizar un ascenso utilizando un método quasi-Newton [4].

4

Herramienta de optimización Bayesiana

El desarrollo de la herramienta que utiliza optimización Bayesiana, cuyo marco teórico se abordó en el capítulo 3, ha sido realizada en el lenguaje de programación Python. Python es un lenguaje interpretado de alto nivel, cuya sintaxis es muy parecida a la del pseudocódigo. Además, es una de las opciones preferidas tanto por científicos como por ingenieros para la computación científica, debido a la sencillez con la que permite codificar y mantener programas [7]. También cuenta con numerosas extensiones, como NumPy, SciPy, Cython o Matplotlib, los cuales aumentan sus ya amplios usos. Las extensiones NumPy y SciPy fueron utilizadas en el desarrollo de la herramienta [8] [9], y Matplotlib para crear todas las gráficas de este trabajo [10].

4.1. GPy

GPy es un framework de procesos Gaussianos escrito en Python [11]. A fin de construir un proceso Gaussiano, GPy cuenta con la clase `kern` que implementa los dos kernels utilizados en este trabajo, RBF y Matérn52, además de muchos otros. Un ejemplo de creación y representación de un kernel se puede ver a continuación

Código 4.1: Ejemplo de la creación de un kernel para: 1 dimensión de entrada, varianza=1.0 y $length - scale$=1.0. Utilizando el framework con GPy

```
1 import GPy
2 kernel = GPy.kern.RBF(input_dim=1, variance=1., lengthscale=1.)
3 kernel.plot()
```

De forma similar al código escrito en 4.1, se escribió el que genera las gráficas de las figuras 2.3 y 2.4. Esta clase también cuenta con el método `K(X1, X2)`, que permite calcular el kernel entre `X1` y `X2`, y fue el método que se utilizó para crear las figuras 2.6 y 2.5. Sin embargo, se debe tener en cuenta que no es necesario trabajar directamente esta clase, a menos que el objetivo consista en trabajar directamente con kernels, ya que los modelos que implementa el framework ya se encargan de esto. Entre estos modelos se encuentra `GPRegression`, que es el que ha sido utilizado en este trabajo. Una

vez se ha creado el $\mathcal{GP}$, este es capaz de llevar a cabo el cálculo de la distribución predictiva (basada en los datos previamente proporcionados), así como el ajuste de los hiperparámetros (mediante la maximización de la verosimilitud marginal). Un ejemplo de esto se observa en el siguiente ejemplo de código

Código 4.2: Ejemplo de la creación y optimización de un $\mathcal{GP}$ con kernel RBF. La función subyacente es $\sin(x)$. Utilizando el framework con GPy

```
1  import GPy
2  import numpy as np
3
4  np.random.seed(0) # Opcional
5
6  def underlingFunction(x):
7      '''Funcion subyacente sin(x)'''
8      return np.sin(x)
9
10 X_input = np.random.uniform(-3.,3.,(5,1))
11 Y = underlingFunction(X_input) + np.random.randn()*0.05
12
13 kernel = GPy.kern.RBF(input_dim=1, variance=1., lengthscale=1.)
14 model = GPy.models.GPRegression(X_input, Y, kernel)
15 model.optimize()
16
17 x_test = np.atleast_2d(1)
18 model.predict_noiseless(x_test) # f_pred = 0.79835093, sigma = 2.35066888e-06
```

Nótese que se ha utilizado el método `predict_noiseless()`, ya que el objetivo es calcular $\mathbf{f}_*$ y no $\mathbf{y}_*$, como se mencionó anteriormente (sección 2.2.1).

4.2. BOTool

La herramienta de optimización bayesiana que se ha desarrollado en este trabajo es BOTool, que está formada por cuatro clases:

Dataset Implementa la extracción de datos de un archivo pasado por parámetro. Los atributos deben estar separados por comas, poniendo al final la clase, y cada fila es una muestra distinta. Sus parámetros de entrada son:

`directory` : directorio y nombre del archivo donde se encuentra el dataset

`name` : (opcional) nombre del dataset

UnderlyingFunction Permite tanto crear como utilizar una de las funciones subyacentes ya creadas, las cuales son: Branin, Hartmann 3, SVM y MLP, que serán comentadas en el capítulo 5. Los parámetros de entrada se encuentran a continuación:

`function` : *función subyacente* que será evaluada, se puede especificar una función propia o utilizar una de las ya programadas. Es `None` por defecto.

`name` : nombre de la función. Si `function=None`, entonces deberá ser "Branin", "Hartmann_3", "SVM" o "MLP".

`bounds` : límites en los que se evaluará la función.

`minValue` : (opcional) valor mínimo conocido de la función dentro de `bounds`. Por defecto es `1e-3`.

`dataset` : objeto de la clase Dataset, que es necesario para las funciones que son clasificadores (SVM y MLP).

AcquisitionFunction Implementa las funciones de adquisición EI, LCB y Random. Las ecuaciones que definen EI y LCB son 3.4 y 3.5, respectivamente, mientras que Random simplemente devuelve un valor aleatorio. Esta clase cuenta con los siguientes parámetros de entrada:

`xi_param` : parámetro ξ de la expresión matemática de EI

`kappa` : parámetro ν de la expresión matemática de LCB

BayesianOptimizer Permite utilizar el algoritmo de optimización bayesiana especificado en 3.1. Los parámetros con los que cuenta, y las modificaciones que provocan en su funcionamiento, vienen especificadas a continuación:

`kernel` : debe ser una función kernel de GPy, la cual será el kernel que utilice `GPRegression`.

`acquisitionFunction` : objeto de la clase AdquisitionFunction con la función de adquisición a utilizar.

`underlyingFunction` : objeto de la clase UnderlyngFunction con la $función\ subyacente$ a optimizar.

`directory` : (opcional) directorio donde se guardarán los resultados en caso de que no valga `None`, que es su valor por defecto.

`directoryImgs` : (opcional) directorio donde se guardarán las gráficas de los resultados en caso de que no valga `None`, que es su valor por defecto. Nótese que esto solo será posible si la $función\ subyacente$ tiene 1 o 2 dimensiones

`numIniEvaluations` : (opcional) número de evaluaciones iniciales antes de construir el $\mathcal{GP}$. Esto ayuda a evitar el sobreaprendizaje, tal como se dijo en 2.4. Su valor por defecto es `5`.

`numEvaluations` : (opcional) número de evaluaciones que se realizarán utilizando el algoritmo de optimización Su valor por defecto es `10`.

`seed` : (opcional) semilla que se utilizará para la generación de número aleatorios. Su valor por defecto es `None`.

Además, cuenta con un objeto de su clase interna `Results`, en el cual se guardan los resultados de las observaciones realizadas.

Un ejemplo de la utilización conjunta de estas cuatro clases se puede ver en el código 4.3.

Código 4.3: Ejemplo de minimización de la *función subyacente* `myFunction`, por medio de optimización Bayesiana. También se muestra cómo crear un Dataset, una la *función subyacente* y una función de adquisición.

```
1  import GPy
2  import numpy as np
3  from sklearn.svm import SVC
4  from sklearn.model_selection import train_test_split
5
6  from AcquisitionFunction import AcquisitionFunction
7  from BayesianOptimizer import BayesianOptimizer
8  from UnderlyingFunction import UnderlyingFunction
9  from Dataset import Dataset
10
11 dataset = Dataset(name="myDataset", directory="datasets/ionosphere/ionosphere.data")
12
13 def myFunction(x_input):
14     C = x_input[0]
15     gamma = x_input[1]
16
17     X_train, X_test, y_train, y_test = train_test_split(
18         dataset.getData(), dataset.getTarget(), test_size=0.3, random_state=0)
19
20     clf = SVC(C=C, gamma=gamma, kernel='rbf')
21     clf.fit(X_train, y_train)
22     return 1 -clf.score(X_test, y_test)
23
24
25 bounds = np.array([[0.01, 5], [0.01, 10]])
26 underlyingFunction = UnderlyingFunction(name="myFunction", function=myFunction, bounds=bounds)
27 acquisitionFunction = AcquisitionFunction(name="EI", xi_param=0.01)
28
29 bOptimizer = BayesianOptimizer(kernel=GPy.kern.RBF(input_dim=2, variance=1., lengthscale=1.),
30                                acquisitionFunction=acquisitionFunction,
31                                underlyingFunction=underlyingFunction,
32                                numIniEvaluations=5,
33                                numEvaluations=100,
34                                seed=0)
35
36
37 bOptimizer.optimizeUnderlyingFunction() # x_min = [2.00472511, 2.70353322]
```

5

Experimentos realizados

A fin de comprobar el funcionamiento de la herramienta de optimización Bayesiana desarrollada, se realizaron experimentos con datos sintéticos y con datos reales, dos en cada caso. En todos ellos el objetivo era común: obtener el mínimo de la función subyacente mediante el uso de diferentes estrategias de elección del siguiente valor a observar.

Para evitar el $overfitting$ del que se habló en la sección 2.4, en los experimentos cuyo espacio de entradas es de 2 dimensiones, se realizaron 5 evaluaciones previas a la aplicación de cualquier estrategia, mientras que en el experimento con 3 dimensiones se llevaron a cabo 30. Esto se realizó de forma que todas las estrategias de cada experimento comenzasen con las mismas observaciones realizadas, y por consiguiente el mismo mejor valor inicial.

En la sección 5.1 se discutirán los experimentos sintéticos, los cuales fueron realizados con las funciones Branin y Hartmann 3 [12] [13], que ya han sido utilizadas anteriormente para probar el funcionamiento de la optimización Bayesiana en [4]. Por otro lado, en la sección 5.2 se utilizó la herramienta, con el propósito de optimizar dos de los hiperparámetros de los clasificadores SVM y MLP, la implementación utilizada de estos clasificadores ha sido tomada de [14].

5.1. Experimentos con datos sintéticos

En el primer experimento sintético se probó con la función Branin que tiene un espacio de entradas de 2 dimensiones, y viene definida por la siguiente expresión matemática

$$f(\mathbf{x})) = (x_2 - \frac{5{,}1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10 \tag{5.1}$$

Como sería infructífero realizar una búsqueda en todo $\mathbb{R}^2$, se fijaron los límites del espacio de entradas a $x_1 \in [-5, 10]$ y $x_2 \in [0, 15]$. En esta función el valor mínimo es: $f(\mathbf{x}_{min}) = 0{,}397887$ y se encuentra en tres puntos: $(-\pi, 12{,}275)$, $(\pi, 2{,}275)$ y $(9{,}42478, 2{,}475)$, dentro de los límites establecidos. La forma de esta función en los límites especificados se puede ver en la figura 5.1.

Por otro lado, en la figura 5.2, se muestra una comparativa de tres estrategias de optimización. La

primera realiza una búsqueda aleatoria, mientras que las otras dos utilizan la función de adquisición EI (3.4). Además una de estas dos últimas utilizaba el kernel RBF y la otra el de Matérn52.

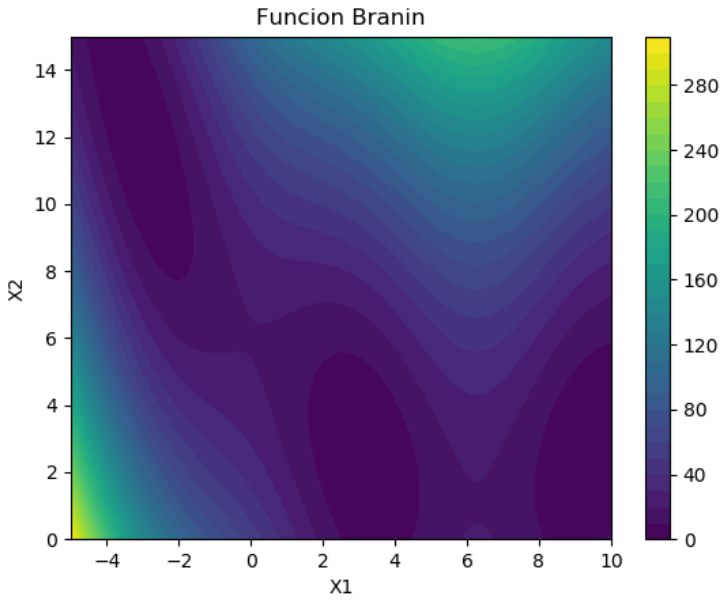


Figura 5.1: Representación en dos dimensiones de la forma de la función Branin. Como se puede ver en hay tres zonas con un azul más oscuro donde se encuentran los mínimos de la función en los límites especificados. [Elaboración propia]

En la gráfica 5.2 se puede apreciar cómo, después de las primeras 50 evaluaciones, las estrategias que realizan sus observaciones basándose en el modelo que construyen obtienen resultados que pasan de 2 a 4 veces mejores entre las 50 y las 100 evaluaciones, mientras que la búsqueda aleatoria apenas obtiene mejoras y se sitúa en torno a -1, en todo este periodo. También es posible comprobar que el $\mathcal{GP}$ con kernel RBF da lugar a valores más cercanos a $f(\mathbf{x}_{min})$ que al utilizar el kernel de Matérn52. Esto puede deberse a que la $función\ objetivo$ tenga una suavidad superior a la que supone Matérn52, y más semejante a la que supone RBF. En la figura 5.3 se puede ver la evolución de la media de un $\mathcal{GP}$ y de la función de adquisición, durante 3 observaciones.

En el caso del experimento con la función Hartmann 3, no será posible mostrar su visualización debido a que su espacio de entradas es tridimensional, como se puede ver en su ecuación

$$f(\mathbf{x}) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2\right) \tag{5.2}$$

donde

$$\alpha = (1{,}0, 1{,}2, 3{,}0, 3{,}2^{\mathsf{T}}), \quad A = \begin{pmatrix} 3{,}0 & 10 & 30 \\ 0{,}1 & 10 & 35 \\ 3{,}0 & 10 & 30 \\ 0{,}1 & 10 & 35 \end{pmatrix}, \quad P = 10^{-4} \begin{pmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{pmatrix}$$

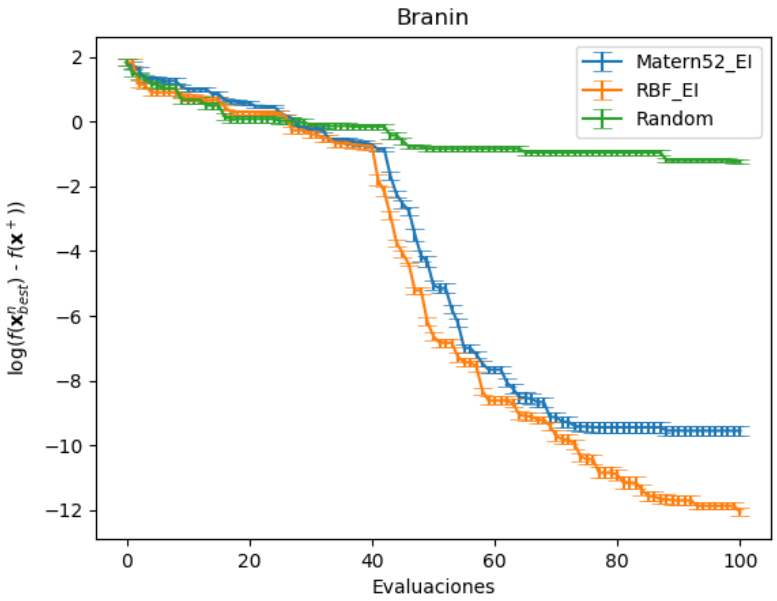


Figura 5.2: Representación del $\log(f(\mathbf{x}^{n}{}_{best}) - f(\mathbf{x}^{+}))$ en cada una de las 100 evaluaciones, donde $\mathbf{x}^{n}{}_{best}$ corresponde al mejor valor encontrado en n evaluaciones ($n = 1, 2, ..., num\_evaluaciones$), y $\mathbf{x}^{+}$ es el mínimo dentro de los límites especificados. La media y el error estándar fueron obtenidas mediante la repetición del experimento 10 veces. [Elaboración propia]

En esta ocasión, los límites que se utilizaron fueron x_1, x_2, $x_3 \in (0, 1)$, cuyo mínimo global se encuentra en $\mathbf{x}_{min} = (0{,}114514, 0{,}555649, 0{,}852547)$, con un valor $f(\mathbf{x}_{min}) = -3{,}86278$.

Se puede apreciar como en la gráfica 5.4, después de las 10 primeras evaluaciones, ya es conveniente utilizar una estrategia basada en el modelo. Además, en diferencia con la gráfica 5.2, aunque Matérn52 obtenga valores más cercanos al mínimo, en las últimas evaluaciones esa diferencia disminuye hasta alcanzar valores inferiores al 5 %. Es posible que la diferencia inicial se produzca debido a que la $función\ subyacente$ supuesta por Matérn52 sea menos suave que RBF. Sin embargo, se puede apreciar que tras 100 evaluaciones, esta diferencia deja de ser significativa.

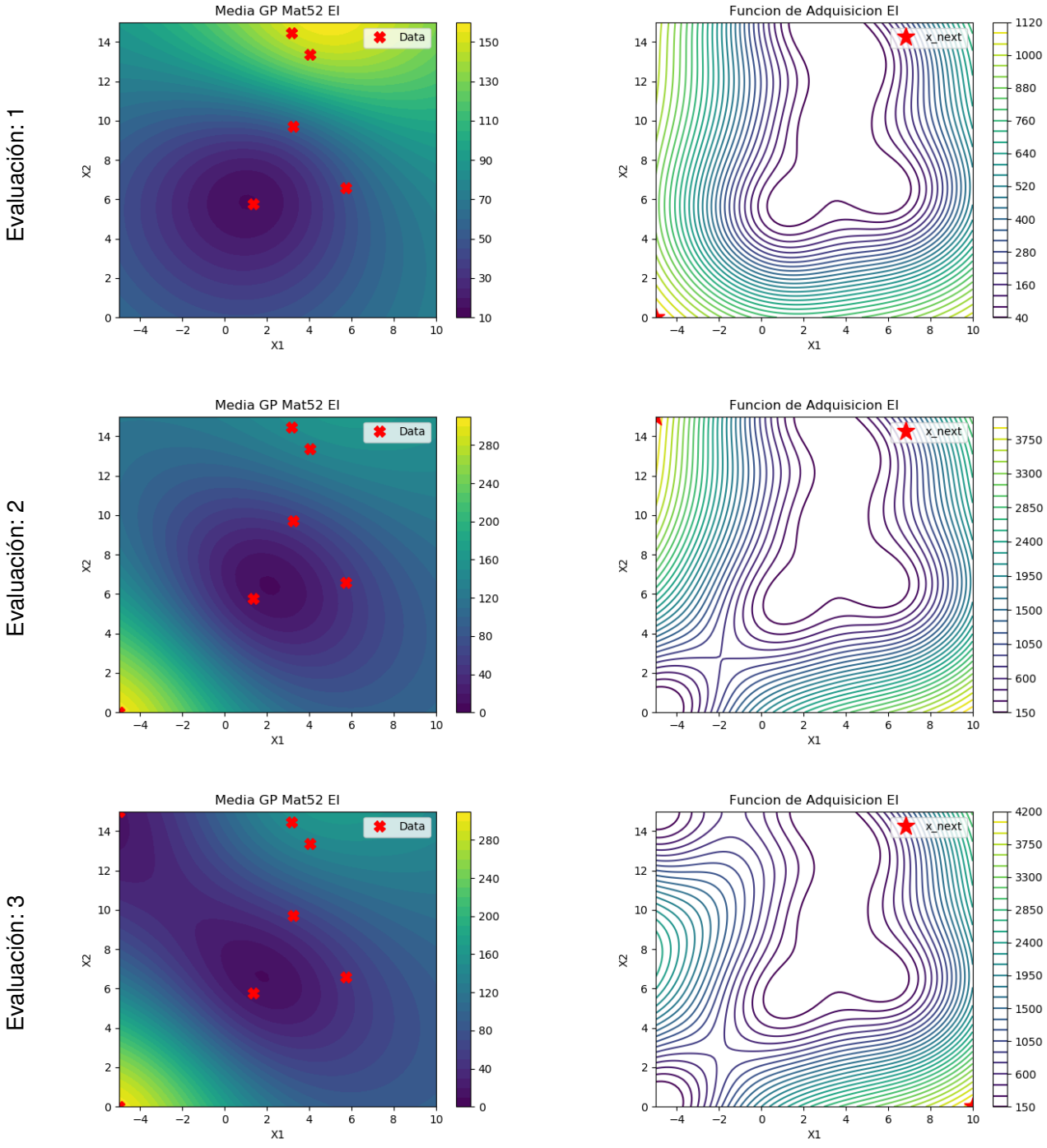


Figura 5.3: Evolución de la media de la distribución predictiva a posteriori de un $\mathcal{GP}$ con kernel Matérn52, y de la función de adquisición EI en la función Branin, durante tres observaciones. [Elaboración propia]

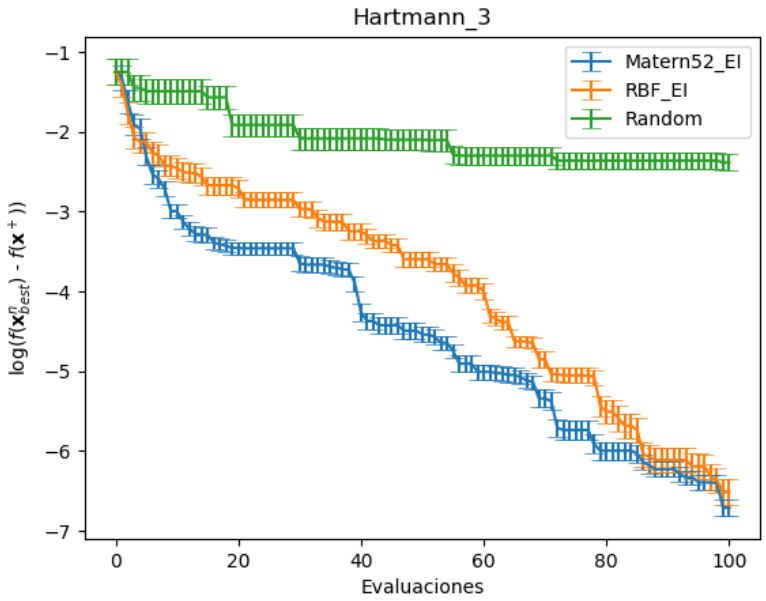


Figura 5.4: Representación del $\log(f(\mathbf{x}^{n}{}_{best}) - f(\mathbf{x}^{+}))$ en cada una de las 100 evaluaciones, donde $\mathbf{x}^{n}{}_{best}$ corresponde al mejor valor encontrado en n evaluaciones ($n = 1, 2, ..., num\_evaluaciones$), y $\mathbf{x}^{+}$ es el mínimo dentro de los límites especificados. La media y el error estándar fueron obtenidas mediante la repetición del experimento 10 veces. [Elaboración propia]

5.2. Experimentos con datos reales

En los dos experimentos reales llevados a cabo, el objetivo era realizar una clasificación a partir de las muestras extraídas del conjunto de datos ionosphere [15]. En ambos experimentos se utilizó validación cruzada con 10 particiones para medir el error del clasificador.

En el primero de los experimentos se variaron los hiperparámetros C y $gamma$ del clasificador SVM donde C define la penalización por clasificar erróneamente un dato, y $gamma$ regula la anchura del kernel (que utiliza el $\mathcal{GP}$ de SVM). Los límites en los que se probaron los hiperparámetros fueron: $C,\ gamma \in [\exp(-5), \exp(5)]$. La figura 5.5 muestra los resultados obtenidos, y en ella es posible comprobar como la diferencia entre la elección del kernel Matérn52, o RBF, no es superior al 20 %, aunque en esta ocasión el resultado de RBF haya sido superior es probable que se deba a la ejecución. Por otro lado, en contraste con los resultados obtenidos en los experimentos sintéticos, la búsqueda aleatoria ha obtenido valores cercanos a los de RBF y Matérn52. Es posible que estos resultados se deban a que el $\mathcal{GP}$ no sea capaz de aproximar correctamente la $función\ objetivo$. Además, teniendo en cuenta que (en Matérn52) el error medio en la evaluación 0, era de $0{,}0626$, cuando solo se habían realizado 5 observaciones aleatorias, y que el error en la evaluación 100, era de $0{,}0457$, se pueda pensar que el margen de mejora era pequeño.

En el segundo experimento, se inicializo un clasificador MLP con una capa de cincuenta neuronas.

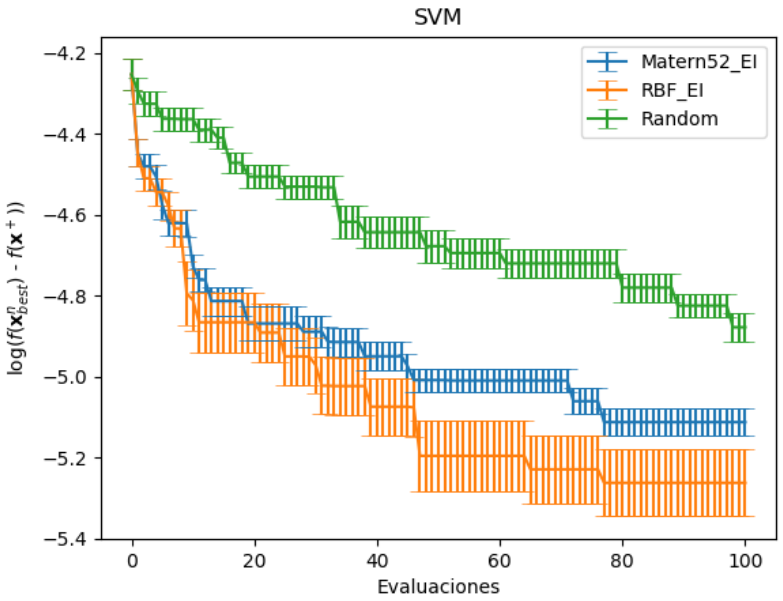


Figura 5.5: Representación del $\log(f(\mathbf{x}^{n}{}_{best}) - f(\mathbf{x}^{+}))$ en cada una de las 100 evaluaciones, donde $\mathbf{x}^{n}{}_{best}$ corresponde al mejor valor encontrado en n evaluaciones ($n = 1, 2, ..., num\_evaluaciones$), y $\mathbf{x}^{+}$ es el mínimo dentro de los límites especificados. La media y el error estándar fueron obtenidas mediante la repetición del experimento 10 veces. [Elaboración propia]

Los hiperparámetros a regular eran $lerning\_rate$ y $alpha$. El $lerning\_rate$ se encarga de controlar el tamaño de las variaciones de los pesos $\mathbf{w}$ en cada iteración, mientras que $alpha$ es el parámetro de penalización L2. Dicho experimento se puede ver en la figura 5.6, en esta ocasión el margen de mejora es menos amplio que en experimento anterior, el recorrido del eje de ordenadas va desde $-4{,}0$ hasta $-4{,}8$. Además, se puede apreciar como la búsqueda aleatoria ha dado unos resultados superiores a los de RBF en casi todo su recorrido y parejos a los de Matérn52. Por otro lado, RBF no ha experimentado ninguna mejora desde la evaluación 20, probablemente debido a que el modelo que aproxima de la $función\ subyacente$ sea incorrecto.

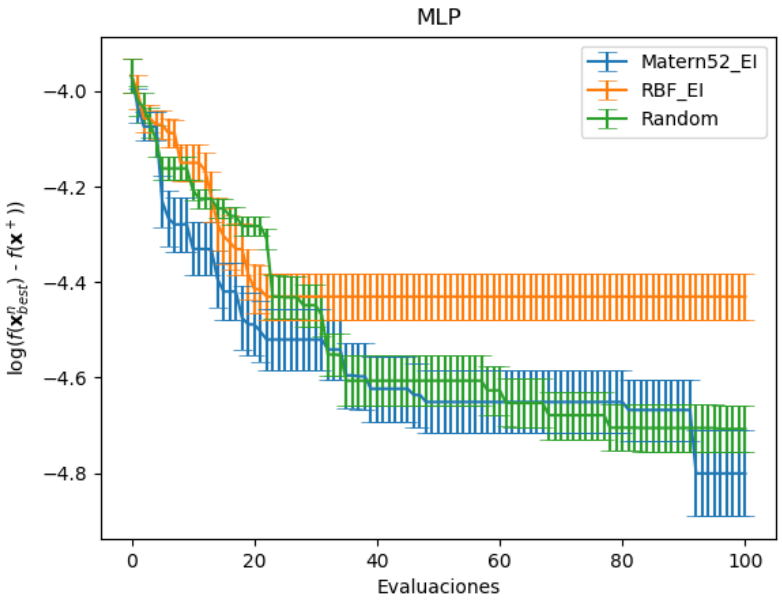


Figura 5.6: Representación del $\log(f(\mathbf{x}^{n}{}_{best}) - f(\mathbf{x}^{+}))$ en cada una de las 100 evaluaciones, donde $\mathbf{x}^{n}{}_{best}$ corresponde al mejor valor encontrado en n evaluaciones ($n = 1, 2, ..., num\_evaluaciones$), y $\mathbf{x}^{+}$ es el mínimo dentro de los límites especificados. La media y el error estándar fueron obtenidas mediante la repetición del experimento 10 veces. [Elaboración propia]

6

Conclusiones y trabajo futuro

6.1. Conclusiones

En este trabajo se ha mostrado como mediante la optimización Bayesiana es posible minimizar el número de evaluaciones a efectuar. A fin de lograr este propósito, la optimización Bayesiana cuenta con un modelo aproximado de la $función\ objetivo$, cuya distribución predictiva puede ser empleada para guiar el proceso de optimización. En cada iteración este proceso realiza una evaluación en el lugar donde la utilidad esperada es máxima, a fin de lograr minimizar el número de iteraciones a realizar. El modelo que generalmente es usado para aproximar la $función\ objetivo$ son los procesos Gaussianos, los cuales, permiten calcular la distribución predictiva a posteriori, condicionando la creencia a priori a las observaciones. Además, se expuso la importancia de la función de adquisición, puesto que debía efectuar el $trade - off$ entre explotación y exploración, al guiar la búsqueda.

Una vez que se supo el marco teórico de la optimización Bayesiana, se realizó el desarrollo de una herramienta que implementase dicha optimización, cuya prueba requirió de dos experimentos, tanto para datos sintéticos como para datos reales. En el primer grupo de estos experimentos se observó que el desempeño de la herramienta era el esperado, dado que la aplicación de técnicas de optimización basadas en el modelo daba resultados superiores frente a las que no lo utilizaban. Sin embargo, en algunos experimentos el uso de técnicas basadas no supone mejoras respecto a la búsqueda aleatoria. Es posible que esto se deba a que haya una gran cantidad de configuraciones de hiperparámetros que den resultados similares. Además, es posible que al haberse utilizado dos kernels $estacionarios$, las asunciones que hacen sobre el espacio inexplorado no sean correctas, debido a que no aproximan bien $funciones\ objetivo$ que no presentan la misma suavidad en todo su recorrido. Sin embargo, existen técnicas que se pueden utilizar para resolver este problema [4].

6.2. Trabajo futuro

En este trabajo se ha visto en profundidad una forma de aplicar la optimización Bayesiana, sin embargo, existen multitud de vías posibles por las que continuar el trabajo. Como se dijo en la sección

2.2.2, el método que se ha utilizado para ajustar los hiperparámetros, maximización de la verosimilitud marginal, tiende a provocar $overfitting$ si hay pocos datos, a fin de evitar esto se podrían obtener muestras de los hiperparámetros del $\mathcal{GP}$ utilizando técnicas Markov chain Monte Carlo [16], esto también evitaría realizar las evaluaciones previas.

Por otra parte, el coste de utilizar los procesos Gaussianos viene determinado por la inversión de la matriz K, que es $O(n^3)$, este coste los hace inviables para la optimización Bayesiana en algunos casos, así que se podrían implementar modelos como $Sparse\ Pseudoinput\ Gaussian\ processes$ (SPGPs) o $Sparse\ spectrum\ Gaussian\ processes$ (SSGP), que reducen el coste evitando utilizar directamente los datos [17] [18]. Además, se podrían desarrollar modelos que escalen mejor que los procesos Gaussianos, como $Random\ Forest$ o $Neural\ Networks$ [19] [20].

También se podría utilizar otro tipo de funciones de adquisición, que buscan minimizar la máxima entropía, como $predictive\ entropy\ search$ (PES), que han dado mejores resultados que EI en algunos casos [21].

Bibliografía

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. the MIT Press, Massachusetts Institute of Technology, 2006. Descarga.

[3] B. Matern, *Reports of the Forest Research Institute of Sweden*. Springer-Verlang, 1960.

[4] R. A. B. Shahriari, K. Swersky Z. Wang and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, 2016. Descarga.

[5] E. B. V. M. Cora and N. de Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *Dept. Comput. Sci., Univ. British Columbia, Vancouver, BC, Canada, Tech. Rep. UBC*, 2010. Descarga.

[6] H. J. Kushner, "A new method of locating the maximum of an arbitrary multipeak curve in the presence of noise," *J. Basic Engineering*, vol. 86, pp. 87–106, 1964.

[7] e. P.F. Dubois, "Python: Batteries included," *Computing in Science and Engineering*, vol. 9, 2007. Descarga.

[8] S. C. C. Stefan van der Walt and G. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science and Engineering*, vol. 9, pp. 90–95, 2007.

[9] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–.

[10] J. D. Hunter., "Matplotlib: A 2d graphics environment," *Computing in Science and Engineering*, vol. 13, pp. 2–30, 2011.

[11] GPy, "GPy: A gaussian process framework in python." `http://github.com/SheffieldML/GPy`, 2012–.

[12] F. H. B. Jr., "Widely convergent method of finding multiple solutions of simultaneous nonlinear equations," *IBM Journal of Research and Development*, vol. 16, no. 5, pp. 504–522, 1972.

[13] J. K. Hartman, "Some experiments in global optimization," 1972.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[15] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[16] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *In Advances in Neural Information Processing Systems*, no. 5, pp. 2951–2959, 2012. Descarga.

[17] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," *In Advances in Neural Information Processing Systems*, no. 5, pp. 1257–1264, 2005. Descarga.

[18] M. Lazaro-Gredilla, J. Qui nonero Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal, “Sparse spectrum gaussian process regression,” *The Journal of Machine Learning Research*, pp. 1865–1881, 2010. Descarga.

[19] F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Sequential model-based optimization for general algorithm configuration,” *LION*, pp. 507–523, 2011. Descarga.

[20] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, Prabhat, and R. Adams, “Scalable bayesian optimization using deep neural networks,” *International Conference on Machine Learning*, 2015. Descarga.

[21] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani, “Predictive entropy search for efficient global optimization of black-box functions,” *Advances in Neural Information Processing Systems*, 2014. Descarga.